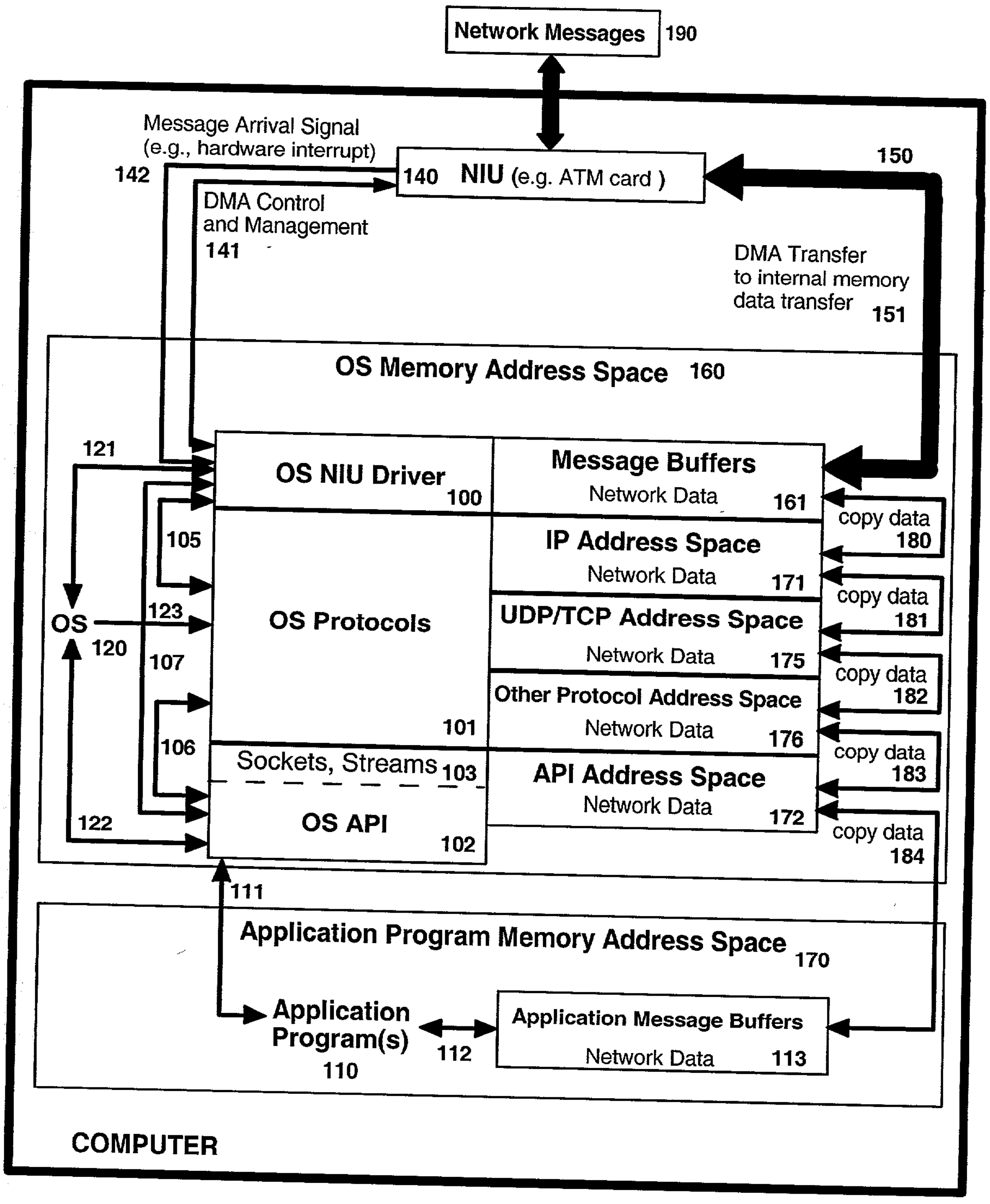
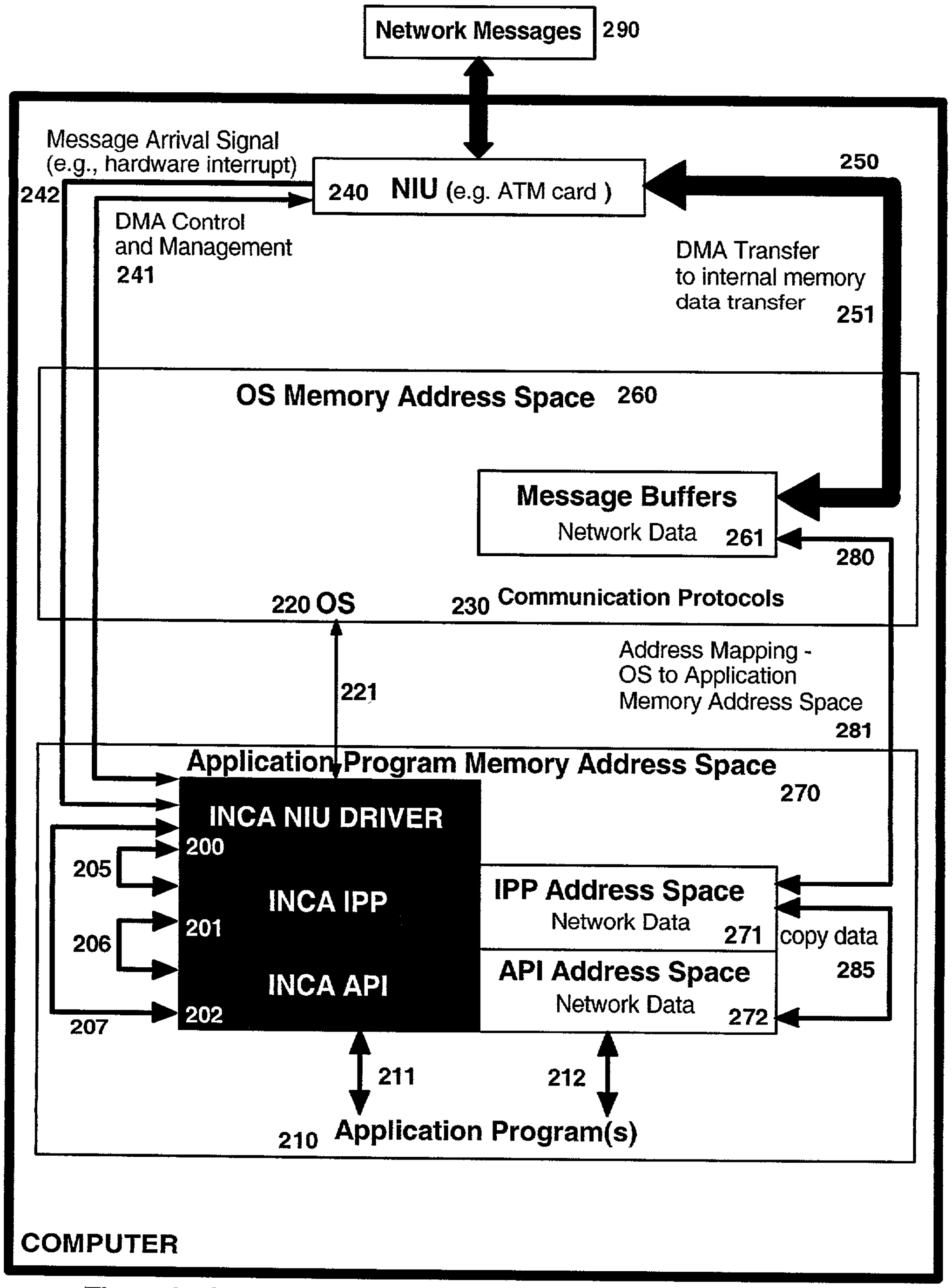

Network Messages 190
Message Arrival Signal (e.g., hardware interrupt)
142
140 NIU (e.g. ATM card)
150
DMA Control and Management 141
DMA Transfer to internal memory data transfer 151
OS Memory Address Space 160
121
OS NIU Driver 100
Message Buffers Network Data 161
copy data 180
105
IP Address Space Network Data 171
copy data 181
OS 120
123
OS Protocols 101
UDP/TCP Address Space Network Data 175
copy data 182
107
Other Protocol Address Space Network Data 176
copy data 183
106
Sockets, Streams 103
OS API 102
API Address Space Network Data 172
copy data 184
122
111
Application Program Memory Address Space 170
Application Program(s) 110
112
Application Message Buffers Network Data 113
COMPUTER

Figure 1. Overview of prior Art Network Communication System



Network Messages 290
Message Arrival Signal (e.g., hardware interrupt)
242
240 NIU (e.g. ATM card)
250
DMA Control and Management 241
DMA Transfer to internal memory data transfer 251
OS Memory Address Space 260
Message Buffers
Network Data 261
280
220 OS
230 Communication Protocols
221
Address Mapping - OS to Application Memory Address Space 281
Application Program Memory Address Space
270
INCA NIU DRIVER
200
205
INCA IPP
201
206
INCA API
202
207
IPP Address Space
Network Data 271
copy data
285
API Address Space
Network Data 272
211
212
210 Application Program(s)
COMPUTER

Figure 2. Overview of the INCA Network Communication System (INCA Integrated into Application)

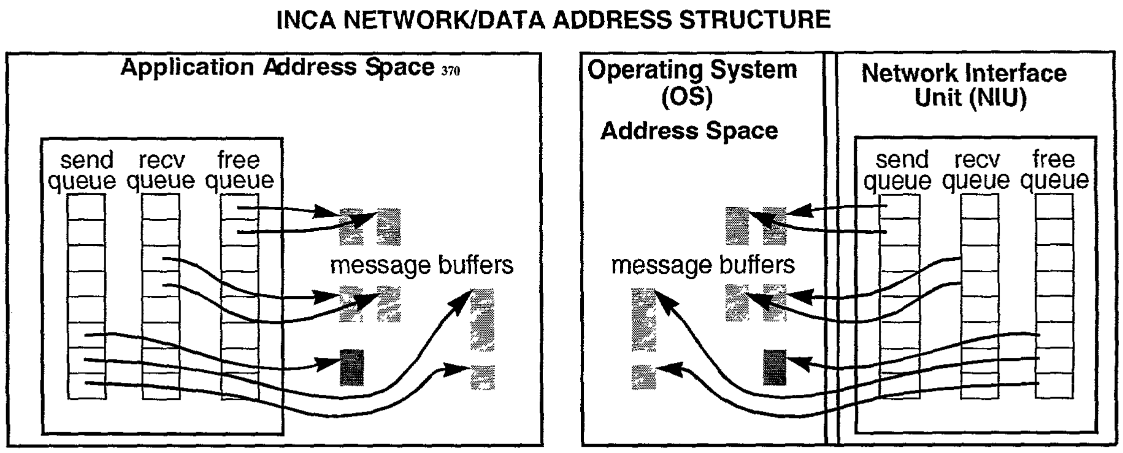

INCA NETWORK/DATA ADDRESS STRUCTURE
Application Address Space 370
send queue
recv queue
free queue
message buffers
Operating System (OS) Address Space
message buffers
Network Interface Unit (NIU)
send queue
recv queue
free queue

Figure 3. INCA Network Data/Mapping Address Data Structure Mechanism

```
for( i = 0; i < 1000; i++ )
    msgData[i]++;                  /* LOAD, ADD, STORE */
for( i = 0; i < 1000; i++ )
    msgData[i] = ~msgData[i];      /* LOAD, COMPLEMENT, STORE */
```

Figure 4a. Two non-IPP For-Loops Examples for typical prior Art multiple Protocol processing Result in a Read (load) and a Write (store) for each Protocol's individual Loop

```
for( i = 0; i < 1000; i++ ){
    temp = msgData[i];             /* LOAD */
    temp++;                        /* ADD */
    temp = ~temp;                  /* COMPLEMENT */
    msgData[i] = temp;             /* STORE */
            }
```

Figure 4b. Examples of INCA's Integrated IPP For-Loops for multiple Protocol processing result in one read (load) and a write (store) for all processed Protocols

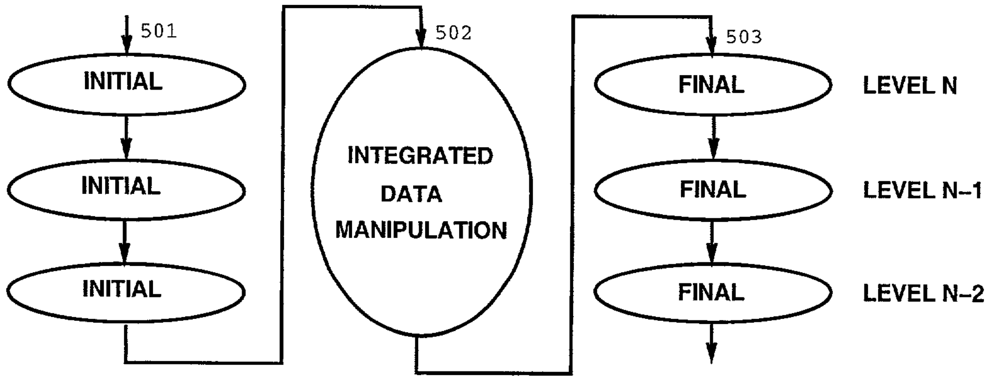


Figure 5. INCA Integrated Protocol Processing (IPP) Execution Stages

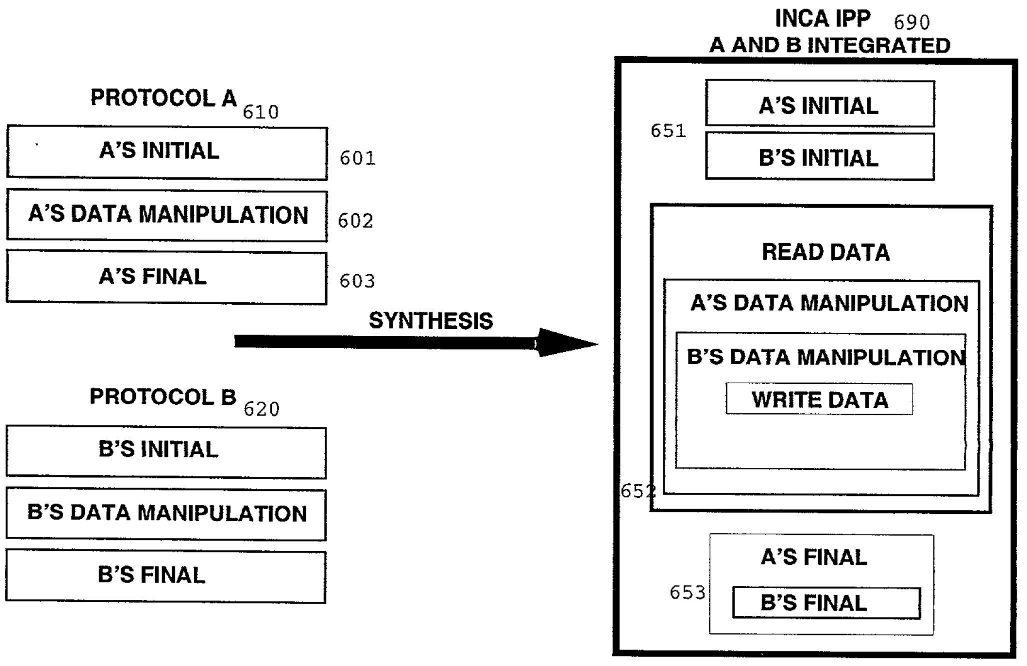


Figure 6. INCA's IPP Method of integrating multiple Protocols

```
inca_t    inca (
                 struct inca_addr local_addr,
                 struct inca_addr remote_addr,
                 int protocol,
                 int family
               )
-----------------------------------------------
int       inca_close (
                     inca_t fd
                   )
-----------------------------------------------
int       inca_connect (
                       inca_t fd
                     )
-----------------------------------------------
int       inca_bind(
                    inca_t fd
                  )
-----------------------------------------------
void      inca_listen (
                      inca_t  fd,
                      int queue_size
                    )
-----------------------------------------------
int       inca_accept (
                       inca_t fd
                     )
-----------------------------------------------
int       inca_send (
                     inca_t fd,
                     char *buffer,
                     int length
                   )
-----------------------------------------------
int       inca_receive (
                         inca_t fd,
                         char *buf,
                         int length
                       )
-----------------------------------------------
void      inca_exit (
                    inca_t  fd
                  )
```

Figure 7. The INCA API Calls and Call Parameters

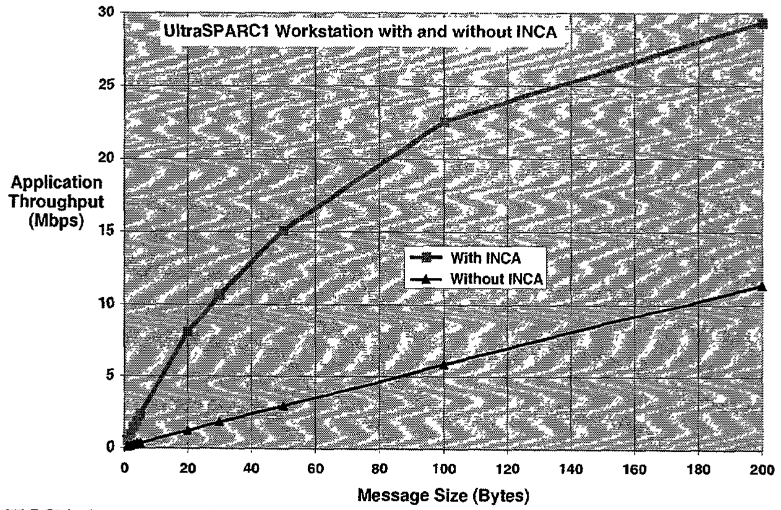


Figure 8. INCA improves Workstation Network Data Handling Performance 260-760%

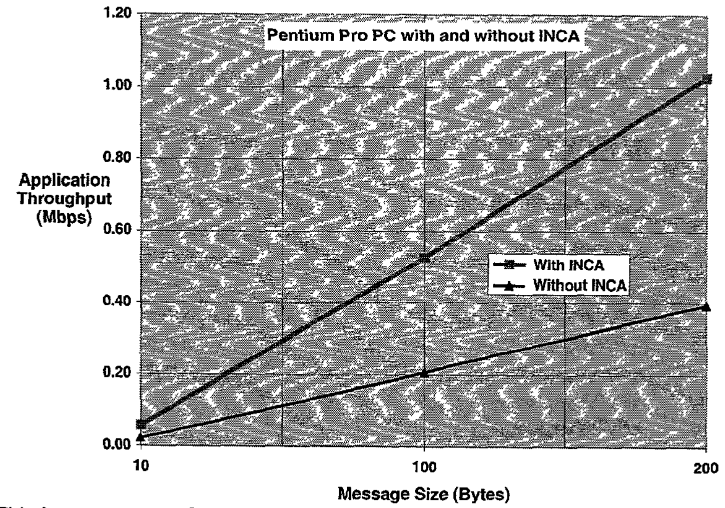


Figure 9. INCA improves PC small Message Network Data Handling Performance by 260-275%

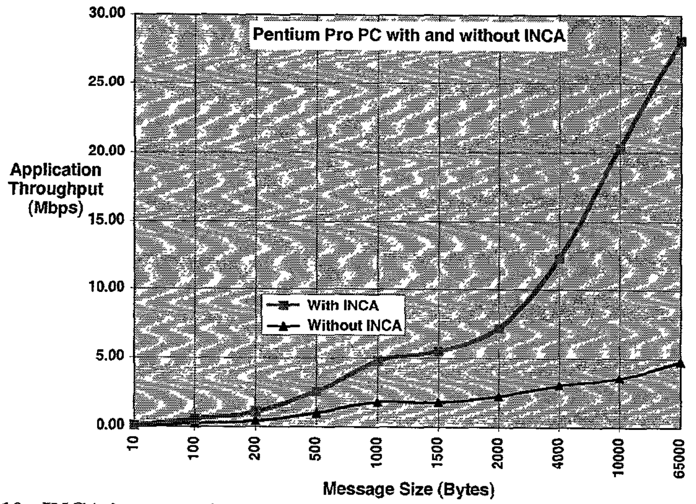
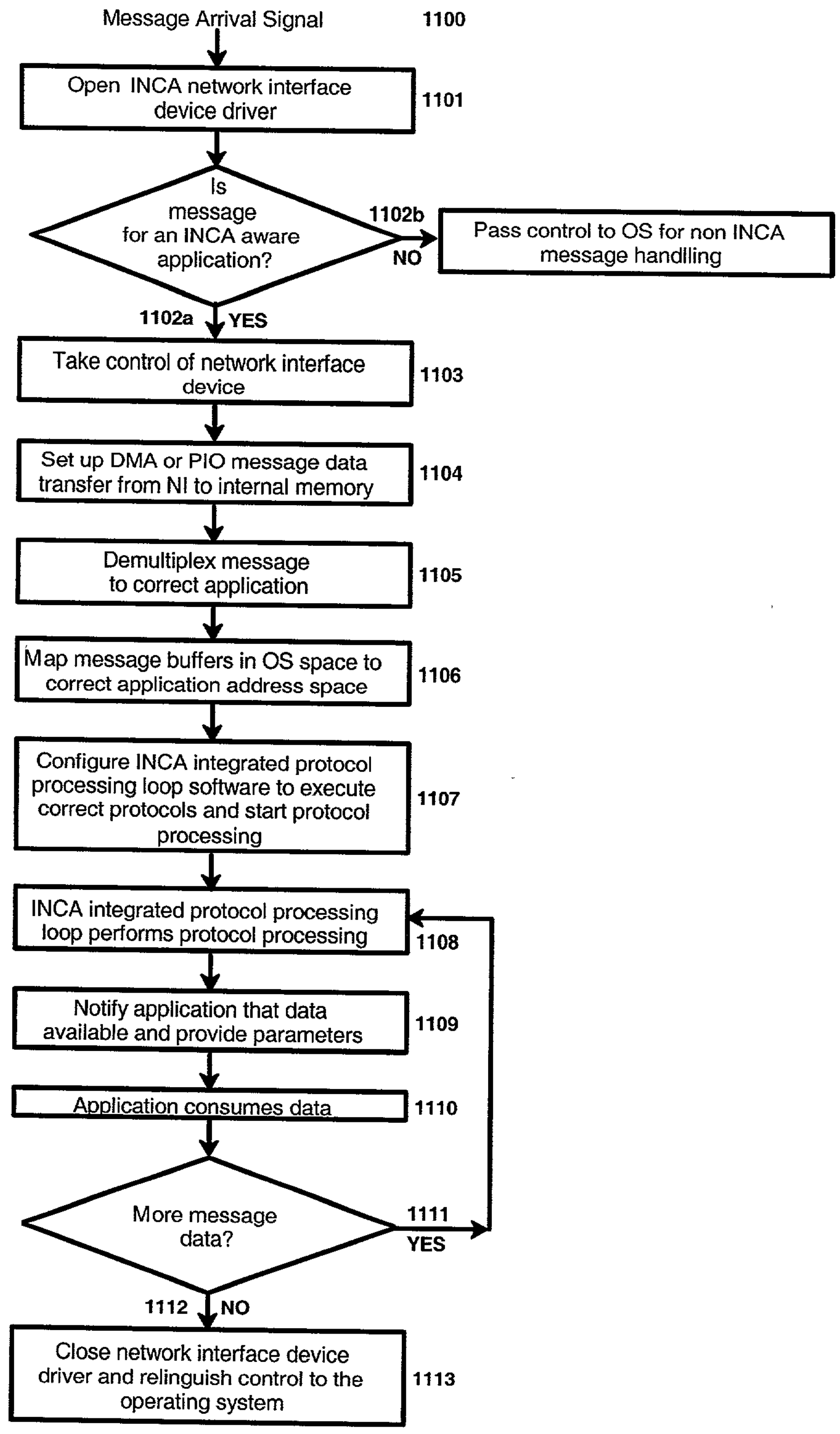


Figure 10. INCA improves PC Network Data Handling Performance up to 590%



Message Arrival Signal
1100
Open INCA network interface device driver
1101
Is message for an INCA aware application?
1102b
NO
Pass control to OS for non INCA message handling
1102a
YES
Take control of network interface device
1103
Set up DMA or PIO message data transfer from NI to internal memory
1104
Demultiplex message to correct application
1105
Map message buffers in OS space to correct application address space
1106
Configure INCA integrated protocol processing loop software to execute correct protocols and start protocol processing
1107
INCA integrated protocol processing loop performs protocol processing
1108
Notify application that data available and provide parameters
1109
Application consumes data
1110
More message data?
1111
YES
1112
NO
Close network interface device driver and relinguish control to the operating system
1113

Figure 11. INCA's Management and Control Flow

```
/* Character/block device ops for INCA Network Driver*/
static struct cb_ops inca_cb_ops = {
                        inca_open,      /* Device Open      */
                        inca_close,     /* Device Close     */
                        nodev,          /* strategy    */
                        nodev,          /* print       */
                        nodev,          /* dump                 */
                        nodev,          /* read             */
                        nodev,          /* write                */
                        inca_ioctl,          /* ioctl               */
                        nodev,               /* devmap      */
                        inca_mmap,           /* mmap                */
                        ddi_segmap,          /* segment map         */
                        nochpoll,
                        nodev,               /* prop_op     */
                        NULL,                /* streams     */
                        (D_NEW|
                            D_MP)                    /*could be D_MP*/
                        };

/* Device operations */
static struct dev_ops inca_ops = {
                          DEVO_REV,
                          0,
                          inca_getinfo,        /* Info */
                          inca_identify,       /* Identify */
                          nulldev,     /* probe        */
                          inca_attach,         /* Device attatch*/
                          inca_detach,         /* Device detatch*/
                          nodev,
                          &inca_cb_ops,        /*Pointer to ops*/
                          (struct bus_ops *)NULL
};
```

Figure 12. INCA Network Driver Entry Points inca_cb_ops Structure

```
for(j=0;j<Length;j+=IPP_UNIUT) {
    /* Read IPP_UNIUT of data, 4/8 byte at a time */
            Data = *input++; / Input Buffer is word aligned */
    if ( IPP_UNIUT == 4) {
            /* Byte Swap */
            Data = ((Data & 0x00FF00FF00) <<8 )|(Data & 0xFF00FF00)>>8);
            /* Check Sum */
            csum += (Data & 0x0000FFFF) + (Data &0xFFFF0000);
    } else {
            /* Byte Swap */
Data = ((Data & 0x00FF00FF00FF00FF) <<8 ) | (Data & 0xFF00FF00FF00FF00)>>8);
            /* Check Sum */
            csum += (Data & 0x000000000000FFFF) + (Data &0x00000000FFFF0000) +
            (Data & 0x0000FFFF00000000) + (Data &0xFFFF000000000000);
    }
```

Figure 13. INCA IPP Example Implementation - Integrating Byte-Swap and Internet Checksumming for 32 and 64 bits

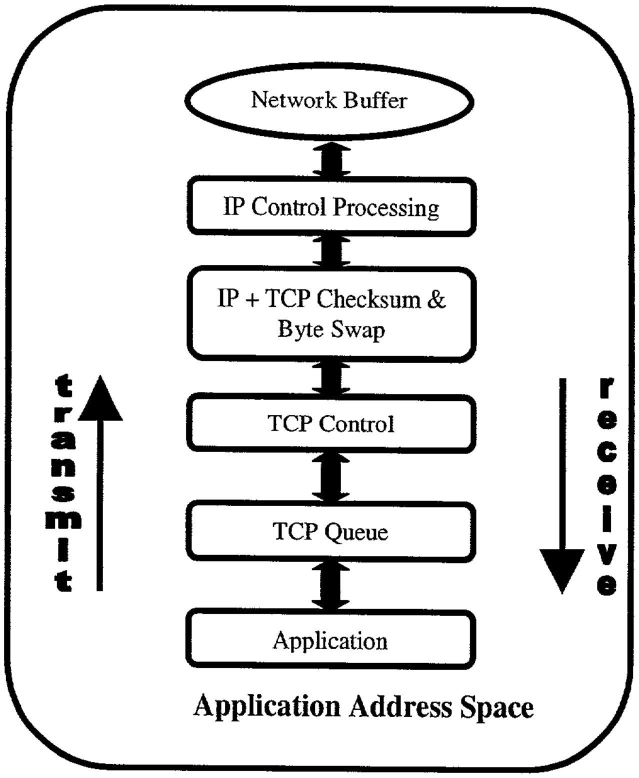


Figure 14. INCA IPP TCP Overview

```
/* Type cast the 4-byte character array of IP address to 4-byte int variable */
if ( *(int *) ip->ip_dst = = *(int *) Connector->ip_src) {
        /* The IP Address is our address. So proceed. */
        register int ip_csum;
        register int udp_csum;

        ip_csum  = *(short *)ip->ip_dst + *(short *)(ip->ip_dst + 2);
        udp_csum =  ip_csum;
        /* Do the rest of the Processing */
}
```

Figure 15. INCA IPP Example of Exploiting Locality with Checksum and Control Processing Integration

```
static int  tcpsend(int tcbnum, Bool rexmt) {
            struct      tcb     *ptcb = &tcbtab[tcbnum];
            struct      ep      *pep;
            struct      ip      *pip;
            struct      tcp     *ptcp;
            char        *pch;
            char *tmp;
            int         i, datalen, tocopy, off, newdata;
            pep = (struct ep *)inca_tx_alloc(sizeof(struct ep));/* Allocate Aligned
                                                                INCA Memory */
            if (pep == (struct ep *)SYSERR)
                  return SYSERR;        /* Allocation Failed */
            pip = (struct ip *)pep->ep_data;     /* Typecast to IP */
           /* INTEGRATING CHECKSUMMING + DATA PROCESSING
              FOR IP AND TCP
           */
            *(int *)pip->ip_src = *(int *)ptcb->tcb_lip;
            ptcb->con->ip_csum += ((*(int *)pip->ip_src & 0xFF00) >> 16) +
                                                  *(int *)pip->ip_src & 0x00FF;
            *(int *)pip->ip_dst = *(int *)ptcb->tcb_rip;
            ptcb->con->ip_csum += ((*(int *)pip->ip_dst & 0xFF00) >> 16) +
                                                  *(int *)pip->ip_dst & 0x00FF;
            ptcp->tcp_sport = ptcb->tcb_lport;
            ptcp->tcp_dport = ptcb->tcb_rport;

            ptcb->con->tcp_csum += ((*(int *)&ptcp->tcp_sport & 0xFF00) >>16) +
                                    *(int *)&ptcp->tcp_sport & 0x00FF;
           /* Continue the TCP send processing */
}
```

Figure 16. INCA IPP Integrating TCP + IP Checksumming with Header Creation for Maximum Locality

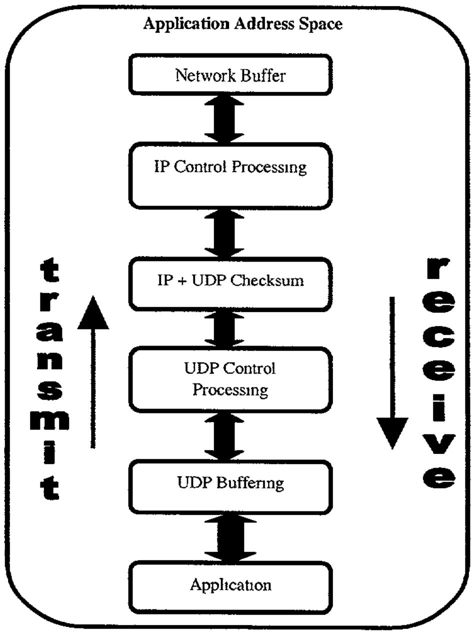


Figure 17. INCA IPP UDP Overview

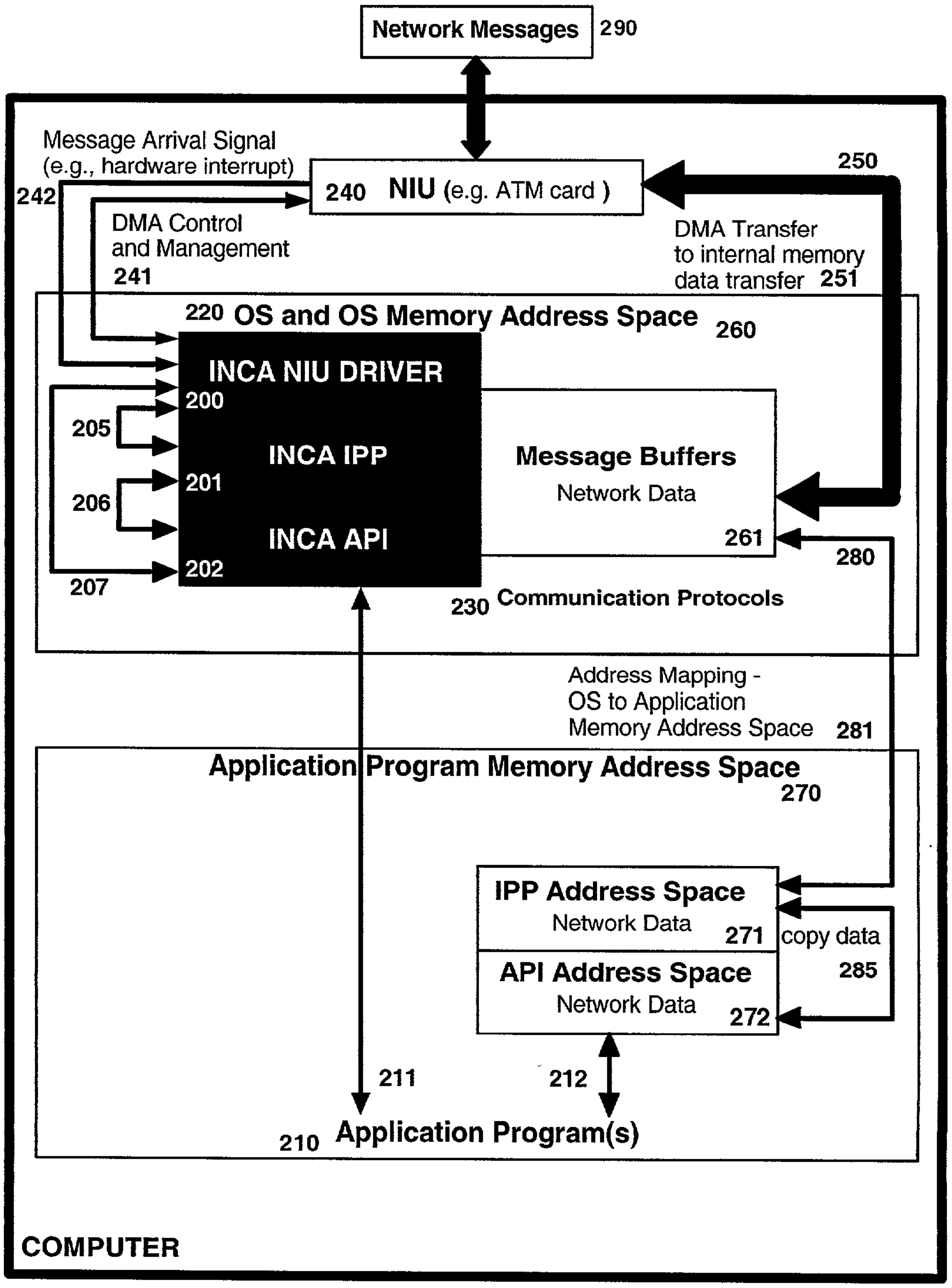

Network Messages 290
Message Arrival Signal (e.g., hardware interrupt)
242
240 NIU (e.g. ATM card)
250
DMA Control and Management 241
DMA Transfer to internal memory data transfer 251
220 OS and OS Memory Address Space 260
INCA NIU DRIVER 200
INCA IPP 201
INCA API 202
205
206
207
Message Buffers
Network Data
261
280
230 Communication Protocols
Address Mapping - OS to Application Memory Address Space 281
Application Program Memory Address Space 270
IPP Address Space
Network Data 271
copy data 285
API Address Space
Network Data 272
211
212
210 Application Program(s)
COMPUTER

Figure 18. INCA Network Communication System Integrated into the OS